Image Morphing Using Deformable Surfaces

Seung-Yong Lee*, Kyung-Yong Chwa*, James Hahn**,
and Sung Yong Shin*

*Department of Computer Science
Korea Advanced Institute of Science and Technology
**Department of EE & CS
The George Washington University

Abstract

This paper presents a new image morphing technique using deformable surfaces. Drawbacks of previous techniques are overcome by a physically-based approach which provides an intuitive model for a warp. A warp is derived by two deformable surfaces which specify horizontal and vertical displacements of points on an image. This paper also considers the control of transition behavior in a metamorphosis sequence. The presented technique separates the transition control from interpolating features making it much easier to use than the previous techniques. The multigrid relaxation method is used to compute a deformable surface for a warp or transition rates. This method makes the presented image morphing technique fast enough for an interactive environment.

1 Introduction

Image morphing deals with the metamorphosis of an image to another image. The metamorphosis generates a sequence of inbetween images in which an image gradually changes into another image over time. Image morphing techniques have been widely used in creating special effects for television commercials, music videos such as Michael Jackson's *Black or White*[1], and movies such as *Willow* and *Indiana Jones and the Last Crusade*[2].

The problem of image morphing is basically to generate an inbetween image from two given images[1]. A natural inbetween image can be derived by properly interpolating the positions of features between two images and their shapes and colors. An image morphing technique first establishes the correspondence of features between two images. The correspondence is then used to compute warps for the images so that the distorted images match the positions and shapes of features. A cross-dissolve of colors at each pixel of the distorted images finally gives an inbetween image.

A warp is a two-dimensional geometric transformation and generates a distorted image when applied to an image. The most difficult part of image morphing is obtaining a warp which provides a necessary distortion of an image. A warp is usually derived from the correspondence of features specified by an animator. Therefore, an image morphing technique must be convenient in specifying features and show a predictable distortion which reflects the feature correspondence.

In mesh warping[2], features are specified by a nonuniform control mesh, and a warp is computed by a spline interpolation. Nishita *et al.*[3] also used a nonuniform control mesh to specify features and computed a warp using a two-dimensional free form deformation and the Bézier clipping. Feature-based morphing[1] specifies features with a set of line segments and computes a warp by taking weighted average of the influences of line segments.

Mesh warping including Nishita's method shows a good distortion behavior but has a critical drawback in specifying features. They always require a control mesh on an image while its features may have an arbitrary structure. Feature-based morphing gives an easy-to-use and expressive method in specifying features but suffers from unexpected distortions referred to as ghosts[1]. A warp computed by feature-based morphing does not interpolate the variation of features, but approximates them as exemplified in Section 5. This prevents an animator from realizing a precise

warp required for a complex metamorphosis. Furthermore, the computation time for a warp is proportional to the number of mesh points or line segments in these techniques. This is disadvantageous when a complicated feature set must be used.

These drawbacks can be overcome by a physically-based approach which provides an intuitive model for a warp. Consider an image printed on a sheet of rubber. When several points on the sheet are moved, the deformation of the sheet makes the image appear distorted. The distorted image conforms to the displacements of points and shows proper distortions between moved points. If we specify the features on an image with a set of points, the displacements of points can be derived from the correspondence of features between images. There have been a number of results[4, 5, 6] in flexible object modeling that give concrete theory and various techniques for developing this approach.

This paper takes the rubber image model and considers horizontal and vertical deformation of the sheet, independently. The deformation problem then reduces to deriving two deformable surfaces which specify horizontal and vertical displacements of points on the sheet. The resulting surfaces are two and a half dimensional and can be represented by explicit functions, $z = f(x, y)$. The requirements for a surface are represented by energy terms, and the surface is computed by minimizing the sum of these terms. The multigrid relaxation method[7] is then used to derive a numerical solution, showing a great enhancement in computation time compared to the conventional relaxation schemes.

Another interesting but not yet fully investigated problem of image morphing is the control of transition behavior in a metamorphosis sequence. In generating an inbetween image, the rate of transition is usually applied uniformly over all points on the image. This results in an animation in which the entire image changes synchronously to another image. If we could control the transition rates on different parts of an inbetween image independently, a more interesting animation could be obtained.

Mesh warping and Nishita's method control the transition behavior with a mesh used for specifying features. Mesh warping assigns a transition curve for each point of the mesh, and the curves determine the transition rate in interpolating the position of each point. Nishita's method specifies the speed of transformation by a Bézier function defined on the mesh.

This paper controls the transition behavior by specifying transition curves for several points on an image. These points are not necessarily the same as those used for specifying features. Transition rates on an inbetween image are derived from the curves by constructing a deformable surface. This approach separates the control of transition from interpolating features making it much easier to use than the previous techniques.

2 Deformable surface construction

The construction of a smooth surface which interpolates a set of scattered points has been investigated in computer vision to solve the visual surface reconstruction problem. Grimson[8] first applied the thin plate surface model[9] to the problem and used the conjugate gradient method for a numerical solution. Terzopoulos extended the result to accommodate discontinuities[10] and presented a multilevel algorithm for quickly solving a hierarchy of discrete problems[11]. We adopt the thin plate surface model and modify the model to control the influence of deformation around feature points. In obtaining a numerical solution, we use the multigrid relaxation method to accelerate the convergence.

2.1 The surface model

Consider a rectangular thin plate placed on the xy-plane and a set of points scattered in the space. Let each point be connected to the plate by an ideal vertical spring of length zero. The stretching of springs gives rise to a potential energy which reduces as the plate is deformed toward the points. When the plate is deformed, the elasticity of the plate prevents a strict bending, resulting in a smooth shape. If the plate is stuck to the xy-plane with a glue, the deformation caused by a spring is localized depending on the viscosity of the glue.

This plate model provides an intuitive interpretation of the surface construction problem. The problem takes a set P of points as the input and finds a C^1-continuous surface which interpolates the points in P. A surface represents a shape of the plate which can be specified by an explicit function $z = f(x, y)$. The interpolation constraint can be forced by minimizing the potential energy due to springs,

$$E_I(f) = \beta \sum_{k=1}^{p} (f(x_k, y_k) - z_k)^2. \qquad (1)$$

(x_k, y_k, z_k) is the coordinates of a point in P and p denotes the size of P. β is a positive real spring constant. It controls the tightness of the interpolation constraints.

The elasticity of the plate can be approximated by minimizing the integration of curvature variations

over the plate,

$$E_C(f) = \iint_\Omega [(\frac{\partial^2 f}{\partial x^2})^2 + 2(\frac{\partial^2 f}{\partial x \partial y})^2 + (\frac{\partial^2 f}{\partial y^2})^2] dx dy. \quad (2)$$

Ω denotes the rectangular domain of the xy-plane on which the surface f is defined. Because a strict bending generates an infinite curvature variation, the resulting surface f has continuous first partial derivatives[10]. The area of deformation can be related to the glue between the plate and the xy-plane. When the plate detaches from the xy-plane, the glue gives rise to an energy which can be estimated by

$$E_L(f) = \alpha \iint_\Omega f^2 \, dx dy. \quad (3)$$

The parameter α corresponds to the viscosity of the glue. It controls the resistance of the surface f to detachment from the xy-plane.

Consequently, the desired surface f can be obtained by minimizing the following energy functional, which is the sum of energy terms (1), (2), and (3).

$$E(f) = \frac{1}{2}(E_C(f) + E_L(f) + E_I(f)). \quad (4)$$

When the parameter β is sufficiently large, the constructed surface f strictly interpolates the points in P. A large α permits the surface f to be apart from the xy-plane only in the small neighborhood of a point in P.

The calculus of variations[9] suggests a powerful technique for generating the desired surface. If a surface f minimizes the energy functional (4), the first variational derivative of (4) must vanish all over the domain Ω. That is,

$$\frac{\delta E}{\delta f} = [\frac{\partial^4 f}{\partial x^4} + 2\frac{\partial^4 f}{\partial x^2 \partial y^2} + \frac{\partial^4 f}{\partial y^4}] + \alpha f + \beta(f(x_k, y_k) - z_k) \equiv 0. \quad (5)$$

The last term appears only at (x_k, y_k) in Ω for which (x_k, y_k, z_k) is a point in P. The partial differential equation (5) is known as the Euler-Lagrange equation. Unfortunately, it is in general impossible to obtain an analytic solution for an Euler-Lagrange equation. This suggests a numerical technique applied to a discrete version of the equation.

2.2 Numerical solution

We discretize the domain Ω to a $M \times N$ regular grid and represent a surface f by its values at the nodes on the grid. Each node is indexed by integers (i, j) and f_{ij} denotes the value of f at (i, j). The point set P is converted to a set of (i, j, z_{ij}) which implies that f_{ij} should have the value z_{ij}. The standard finite difference approximation transforms the differential equation (5) into a system of linear equations[12].

The linear system contains MN unknowns and MN equations. If the nodal variables comprising the function f are collected into the MN dimensional vector $\mathbf{f}$, the system may be written in a matrix form,

$$\mathbf{Af} = \mathbf{b}. \quad (6)$$

$\mathbf{A}$ is a $MN \times MN$ matrix which comprises the coefficients multiplied by nodal variables. Due to the local nature of a finite difference discretization, $\mathbf{A}$ has the desired computational property of bandedness. $\mathbf{b}$ is an MN dimensional vector which contains zeros and constraints z_{ij} multiplied by β.

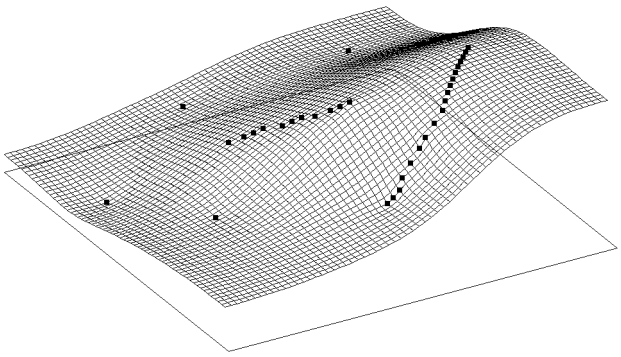
Many types of algorithms have been developed for solving a large banded linear system. Relaxation algorithms such as Jacobi, Gauss-Seidel, or successive-overrelaxation methods exploit the bandedness of the matrix $\mathbf{A}$ to solve the problem efficiently[12]. A major drawback of a relaxation scheme is that it converges slowly in general. The multigrid approach was developed to overcome this drawback and has been actively researched by the numerical analysis community[7, 13].

The multigrid approach applies the ideas of nested iteration and coarse grid correction to a hierarchy of grids[7]. The computational efforts can be estimated in terms of the work unit W which is defined as the amount of computation required for one relaxation on the finest grid. The necessary computation effort is less than $7/2\nu W$, where ν is the number of relaxation on a grid[7]. Because ν is usually small, this is a great enhancement compared to the conventional relaxation schemes. It has been proven that the multigrid approach requires $O(MN)$ operations to reduce the error to the truncation error level[7].

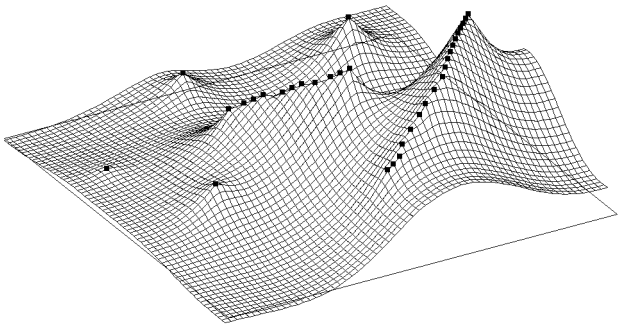
Figure 1 shows surface construction examples in which the finest and coarsest grids are of size 64×64 and 16×16, respectively. In the examples, ν is ten and it takes 0.8 seconds on a SGI 4D/35 for the multigrid relaxation method to generate a surface. Figure 1 also demonstrates how a surface shape can be controlled by the parameter α. When the size of the grid is 512×512, its computation time is 16.9 seconds.

3 Image morphing using deformable surfaces

This section presents an image morphing technique using the surface model given in Section 2. The first subsection defines three subproblems of image



(a) $\alpha = 0$, $\beta = 100$



(b) $\alpha = 0.05$, $\beta = 100$

Figure 1. Surface construction examples: black spots represent the interpolated points

morphing. The following subsections solve the subproblems.

3.1 Problems

When two images I_0 and I_1 are given, the image morphing problem is to find a sequence of inbetween images $I(t)$ such that $I(0) = I_0$ and that $I(1) = I_1$. We assume that time t varies from 0 to 1 when the source image I_0 continuously changes to the destination image I_1. Let W_0 be the warp function which specifies a corresponding point on I_1 for each point on I_0. When applied to I_0, W_0 has to generate the distorted image which matches I_1 in the positions and shapes of features. Its inverse function W_1 distorts I_1 toward I_0. To generate an inbetween image $I(t)$, we derive two warp functions $W_0(t)$ and $W_1(t)$ from W_0 and W_1 by linear interpolation in time t. I_0 and I_1 are then distorted by $W_0(t)$ and $W_1(t)$, resulting in intermediate images $I_0(t)$ and $I_1(t)$, respectively. The corresponding features on I_0 and I_1 have the same positions and shapes on $I_0(t)$ and $I_1(t)$. Finally, $I(t)$ is obtained by cross-dissolving the colors between $I_0(t)$ and $I_1(t)$. That is,

$$W_0(t) = (1-t) \cdot U + t \cdot W_0 \quad (7)$$
$$W_1(t) = t \cdot U + (1-t) \cdot W_1 \quad (8)$$
$$I_0(t) = W_0(t) \circ I_0 \quad (9)$$
$$I_1(t) = W_1(t) \circ I_1 \quad (10)$$
$$I(t) = (1-t) \cdot I_0(t) + t \cdot I_1(t), \quad (11)$$

where U denotes the identity warp function, and $W \circ I$ denotes the application of a warp W to an image I.

Time t in the above formulae controls the transition rate which is uniform on the inbetween image $I(t)$. The rate of transition may be made different from point to point. A transition function T specifies a rate of transition for each point on an image over time. Let T_0 be a transition function defined on the source image I_0. In generating $I(t)$, $T_0(t)$ determines how far each point on I_0 moves to the corresponding point on the destination image I_1. $T_0(t)$ also determines how much the color of each point on I_0 is reflected on the corresponding point on $I(t)$. With the correspondence of points between I_0 and I_1, T_0 can be converted to a transition function T_1 defined on I_1. T_1 specifies the movements and color variations required for points on I_1. To control the movements of points, we replace time t in formulae (7) and (8) with $T_0(t)$ and $T_1(t)$, respectively. To manipulate color transformation, we rearrange formulae (9), (10), and (11). $T_0(t)$ and $T_1(t)$ are then used to attenuate the color intensities of I_0 and I_1. That is,

$$W_0(t) = (1 - T_0(t)) \cdot U + T_0(t) \cdot W_0$$
$$W_1(t) = T_1(t) \cdot U + (1 - T_1(t)) \cdot W_1$$
$$I_0(t) = W_0(t) \circ ((1 - T_0(t)) \cdot I_0)$$
$$I_1(t) = W_1(t) \circ (T_1(t) \cdot I_1)$$
$$I(t) = I_0(t) + I_1(t).$$

The transformation of positions and colors can be independently handled by specifying two transition functions.

To complete the above procedure for image morphing, the following three problems need further investigating.

- How to get the warp function W_0 and its inverse W_1 ?
- How to get the transition function T_0 and its correspondent T_1 ?
- How to apply a warp function to an image ?

3.2 Warp functions

To derive the warp function W_0, an animator is required to specify a set S of point pairs overlaid on the images I_0 and I_1. The set S represents the feature correspondence between I_0 and I_1 which the animator intends to realize. W_0 is composed of two functions X and Y which determine its x- and y-components, respectively.

$$W_0(x, y) = (X(x, y), Y(x, y))$$

We use the x-displacement of each point pair in S to derive the function X. We first define X as

$$X(x, y) = x + \Delta X(x, y).$$

The function ΔX must satisfy the constraints that

$$\Delta X(x_0, y_0) = \Delta x = x_1 - x_0,$$
$$\text{where } p_i = (x_i, y_i),\ i = 0, 1, \text{ and } (p_0, p_1) \in S,$$

because W_0 maps the point p_0 to the point p_1 for each point pair (p_0, p_1) in S. ΔX can be obtained by generating a deformable surface in Section 2, where the set of Δx from S plays the role of interpolation constraints. The function ΔY and Y can be symmetrically derived by the same approach. The resulting warp function W_0 properly reflects the guidance of the animator by interpolating the deviations Δx and Δy.

The warp function W_1 is not necessarily the inverse of W_0 in the mathematical sense. The role of W_1 is to distort the image I_1 so that the point p_1 moves to the point p_0 for each pair (p_0, p_1) in S. Therefore, W_1 can be obtained in the same way as W_0 if we exchange the role of I_0 and I_1 as the source and destination images.

In constructing surfaces for the functions ΔX and ΔY, large β is usually used to get a warp function which exactly moves a point to its correspondent for each point pair in S. Parameter α controls how far the movement of a point is reflected on its neighborhood. Large α makes the function ΔX and ΔY be zero over the domain except on the vicinity of the interpolated points. This results in a warp function which distorts an image only near the feature points.

3.3 Transition functions

To obtain the transition function T_0, an animator selects a set P_0 of points on the image I_0 and specifies a transition curve for each point in P_0. The set P_0 is not necessarily the same as the point set used for warp functions. A transition curve gives the rates of transition over time as shown in Figure 2. The set P_0 and the transition curves represent the metamorphosis scenario designed by the animator. For a given time t, the function $T_0(t)$ can be derived by a method similar to that used for the function X. We first define $T_0(t)$ as

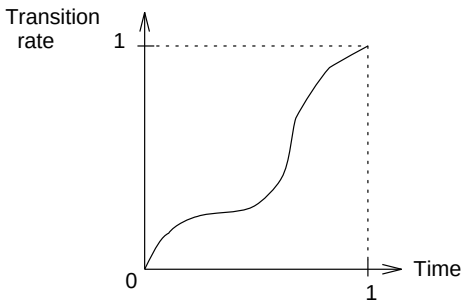


Figure 2. A transition curve

$$T_0(x, y; t) = t + \Delta T(x, y; t).$$

The function ΔT must satisfy the constraint that

$$\Delta T(x, y; t) = \Delta t = C(x, y; t) - t, \text{ for } (x, y) \in P_0,$$

where $C(x, y; t)$ denotes the transition rate at time t computed from the transition curve specified for the point at (x, y). ΔT can then be obtained by constructing a surface which interpolates the set of Δt at points in P_0. The resulting T_0 properly propagates the specified transition curves all over the image I_0.

The transition function T_1 has to specify the transition behavior for the image I_1 which is symmetric to T_0. To obtain T_1, we first apply the warp function W_0 to the point set P_0, getting a point set P_1 on I_1. Then, the transition curve specified for each point in P_0 is assigned to the corresponding point in P_1. With the point set P_1 and the transition curves, T_1 can be computed by the technique used for T_0. Because W_0 provides the correspondence between I_0 and I_1, the resulting T_1 realizes the metamorphosis scenario which is the same as that guided by T_0.

The roles of parameters α and β in computing a transition function is similar to that for a warp function. Parameter α controls the influence area of the transition behavior at a point which is different from the ordinary transition rate t.

When a sequence of inbetween images is generated with a transition function T, a surface should be constructed for determining the function $T(t)$ at each time t. In this case, the solution for a surface is used for the initial solution for the surface at the next time step. Because the surfaces change smoothly with time, this approach provides a good initial solution and reduces the computation time.

3.4 Application of a warp function to an image

Let I be an image and W a warp function which specifies a new position for each point on I. When W is applied to I, each pixel on I is considered as a square that may be transformed into a quadrilateral on a distorted image I'[2]. The resulting quadrilateral from a pixel often straddles several pixels on I' or lies in one pixel. The partial contributions are handled by scaling the intensity of the pixel on I in proportion to the fractional part of the pixel on I' that it covers.

To implement the mapping, we should evaluate a warp function W at each corner of pixels on a given image I. Hence, when the domain of W is discretized for a numerical solution, the size of a grid is chosen as the resolution of I. Once W has been computed on the grid, we can perform the color blending by the blending hardware of a SGI machine[14].

4 Extensions

4.1 General feature control primitives

The relaxations spends most computation time in constructing a surface by the multigrid relaxation method. Because the number of interpolated points is not related to the number of required relaxations, the total computation time remains nearly constant regardless of the number of feature points. This strong merit makes it possible to easily extend the feature control primitives to include line segments and curves.

When a pair of line segments are specified to establish the correspondence of features between images, a discretization of the line segments generates a set of corresponding point pairs. When curves are used to control feature correspondences, the Catmull-Rom spline curves[15] are adopted to interpolate the control points specified by an animator. By properly discretizing the parameter space and computing the points on the curves, we get a set of corresponding points lying on the matching curves. Theses generalized primitives can also be used for controlling transition functions.

4.2 Procedural transition functions

To derive a good metamorphosis, we must match the positions of corresponding features on an inbetween image. Once the positions of corresponding features are aligned, the rate of transition at each point has a freedom in determining its value. Our image morphing technique always generates an inbetween image on which corresponding points on given images match their positions whatever transition function is specified. Therefore, procedural transition functions can be used to generate various interesting inbetween images. For example, the transition function $T_0(x, y; t) = x/x_{max}$ generates an inbetween image which gradually changes from the source image to the destination image when it is scanned from left to right. A procedural transition function defined on an image can be converted to the transition function for the other image by the correspondence of points between the images.

5 Experiments

5.1 Comparison with the feature-based image morphing

The presented image morphing technique can be compared with the feature-based image morphing[1] when line segments are used for specifying features. Figure 3 shows how well our warp function distorts an image to match the variations of features.

Figure 3(a) is the original image in which a letter 'F' lies on a mesh. We overlay line segments on the image and move them to obtain distorted images. Figure 3(b) is generated when the warp function is computed by the feature-based image morphing technique. In the image, the lower bar in 'F' does not shrink in the amount specified by the movements of line segments. The right end of the upper bar in 'F' shows a distortion while the line segment on it is fixed. These abnormal distortions result from that the effects of two or more line segments are blended by simple weighted averaging. In Figure 3(c), the warp function is computed by the technique in this paper. Figure 3(c) exactly reflects the movements of line segments and shows proper distortions over the entire image.

Figure 3(d) and 3(e) show the images obtained when the line segments on Figure 3(a) are moved to distort 'F' to the letter 'T'. The feature-based image morphing technique generates the image in Figure 3(d), which does not show the desired distortion. In the image, the influences of line segments crumble each other and the displacement or rotation of any line segment is not properly reflected. In contrast, the technique presented in this paper gives the exact distortion, as shown in Figure 3(e).

We use a workstation SGI Crimson(R4000) to generate examples in this section. An image in Figure 3 is 420×420 and it takes 12 seconds for the presented technique to generate a distorted image. The feature-based image morphing requires 38 seconds to generate a distorted image. When the number of line segments gets larger, the computation time increases

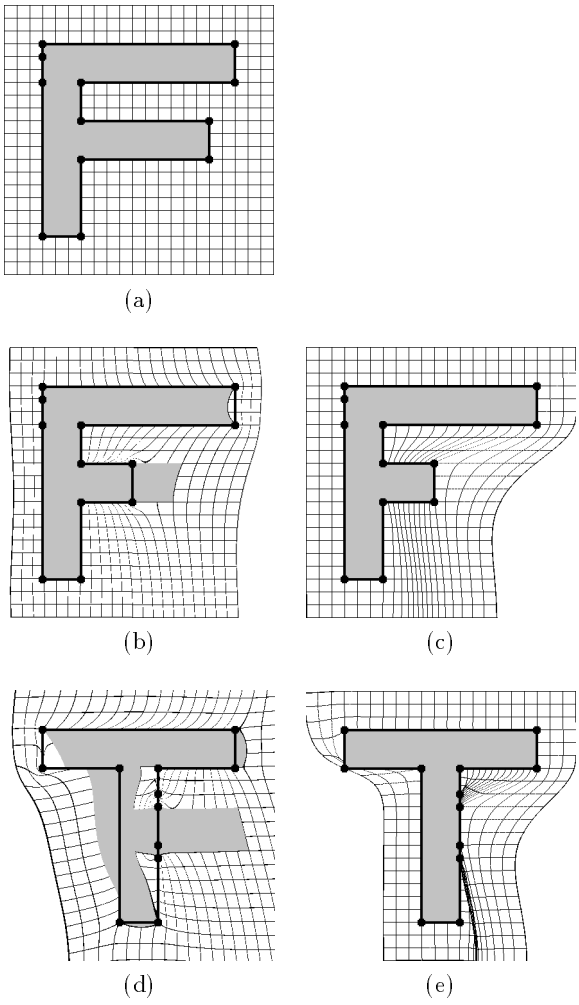


Figure 3. Comparison of warps with feature-based image morphing

in the feature-based image morphing while the time remains nearly constant in the presented technique.

5.2 Examples

Figure 4 shows a metamorphosis example. Figure 4(a) is a face image of the first author. Figure 4(b) is an image of a cat. Figures 4(c) and 4(d) show the feature control primitives overlaid on the images. Figure 4(e) is the middle image on which the same transition rate is applied all over the image. Figure 4(f) is an inbetween image on which transition rates are different part by part. The eyes, nose, and mouth of the image look more like the human face than the remaining parts. Figure 4(g) and 4(h) are examples of applying a procedural transition function, $T_0(x, y; t) = x/x_{max}$ and $T_0(x, y; t) = (\sin(4\pi x/x_{max}) + 1)/2$, respectively.

The resolution of an image in Figure 4 is 400×400 and it takes 4.7 seconds to generate a surface on a 400×400 grid. When generating an inbetween image, two surfaces are generated to compute each of the warp functions W_0 and W_1. Hence, about 18.8 seconds are necessary to generate the image in Figure 4(e). For an inbetween image with transition rates different part by part, two more surfaces are constructed for the transition functions T_0 and T_1. Then, it takes 28.2 seconds to generate the image in Figure 4(f). Except the evaluation of procedural transition functions, the computation required for the images in Figure 4(g) and 4(h) is the same as that for the image in Figure 4(e).

In Figure 5, two different expressions of a person are interpolated to generate a facial animation. The top-left image is the source and the bottom-left is the destination. The inbetween images show that the features are nicely controlled by the presented technique. For example, the motion of the mouth looks natural in the inbetween images. Each image in Figure 5 is 449×423 and it takes 23.6 seconds to generate an inbetween image.

6 Conclusions

This paper presents a new image morphing technique using deformable surfaces. The technique gives freedom to an animator in designing a metamorphosis because features on an image are specified by an arbitrary point set. A warp exactly reflects the correspondence of features between images and shows intuitive distortions among feature points.

We separate the problem of controlling transition behavior from aligning the features and solve the problem with a transition function. An animator can realize a desired transition scenario by specifying transition curves at several points on an image. A procedural transition function can be also used to generate interesting inbetween images.

The multigrid relaxation method is used to compute a surface for a warp or transition function. This method makes the presented image morphing technique fast enough for an interactive environment. Furthermore, the computation time remains nearly constant when the number of control points increases. Hence, the feature control primitives can be extended to include the line segments and spline curves which are more convenient for specifying features.

The most tedious part of image morphing is to establish the correspondence of features between images by an animator. Techniques of computer vision may be employed to automate the problem. An edge de-

tection algorithm can provide important features on images, and an image analysis technique may be used to find the correspondence between detected features.

References

[1] T. Beier and S. Neely. Feature-based image metamorphosis. *Computer Graphics*, 26(2):35–42, 1992.

[2] G. Wolberg. *Digital Image Warping.* IEEE Computer Society Press, 1990.

[3] T. Nishita, T. Fujii, and E. Nakamae. Metamorphosis using Bézier clipping. In *Proceedings of the First Pacific Conference on Computer Graphics and Applications*, pages 162–173, Seoul, Korea, 1993.

[4] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. *Computer Graphics*, 21(4):205–214, 1987.

[5] D. Terzopoulos and K. Fleischer. Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. *Computer Graphics*, 22(4):269–278, 1988.

[6] J. C. Platt and A. H. Barr. Constraint methods for flexible models. *Computer Graphics*, 22(4):279–288, 1988.

[7] W. L. Briggs. *A Multigrid Tutorial.* SIAM, 1987.

[8] W. Grimson. An implementation of a computational theory of visual surface interpolation. *Computer Vision, Graphics, and Image Processing*, 22:39–69, 1983.

[9] I. Gelfand and S. Fomin. *Calculus of Variations.* Prentice-Hall, 1963.

[10] D. Terzopoulos. Regularization of inverse visual problems involving discontinuities. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, PAMI-8(4):413–424, 1986.

[11] D. Terzopoulos. Multilevel computational processes for visual surface reconstruction. *Computer Vision, Graphics, and Image Processing*, 24:52–96, 1983.

[12] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C.* Cambridge University Press, second edition, 1992.

[13] A. Brandt. Multi-level adaptive solutions to boundary-value problems. *Mathematics of Computation*, 31(138):333–390, 1977.

[14] Silicon Graphics Inc. *Graphics Library Programming Guide.*

[15] G. Farin. *Curves and Surfaces for Computer Aided Geometric Design.* Academic Press, second edition, 1990.

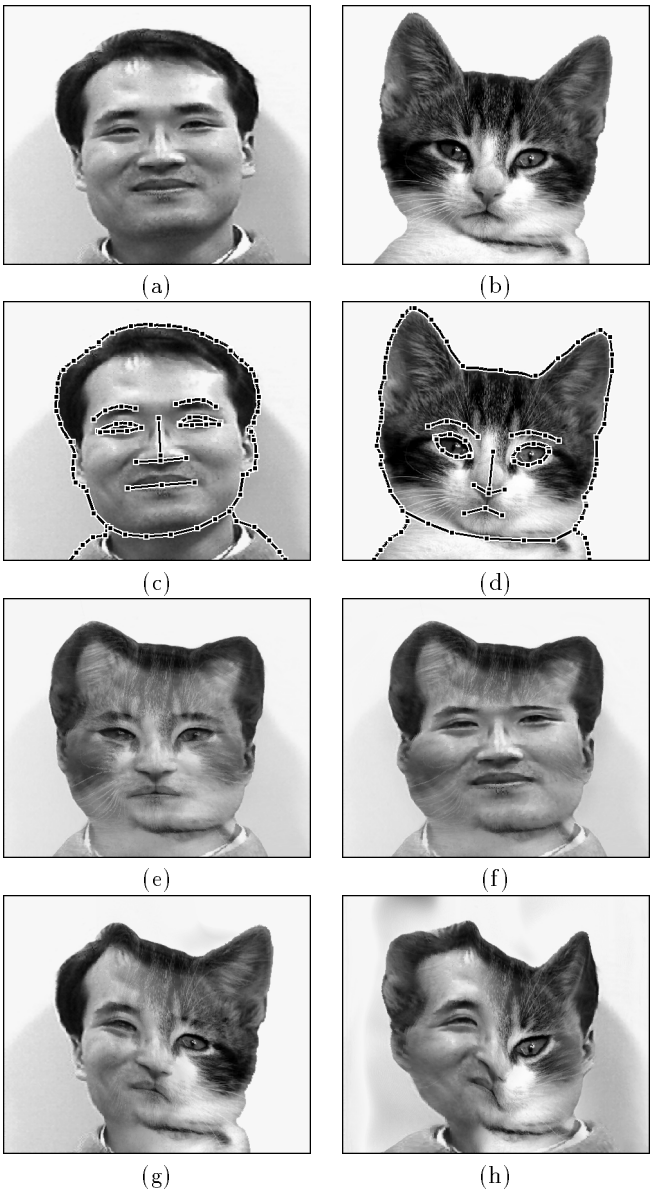


Figure 4: A metamorphosis example: from a person to a cat

Figure 5: A facial animation: the top-left is the source image and the bottom-left is the destination image